



UNITED STATES PATENT AND TRADEMARK OFFICE

107

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/074,779	02/13/2002	Eric M. Dowling	SEARCHP.011C1DV1	7948
27299	7590	04/05/2006	EXAMINER	
GAZDZINSKI & ASSOCIATES 11440 WEST BERNARDO COURT, SUITE 375 SAN DIEGO, CA 92127				HUISMAN, DAVID J
		ART UNIT		PAPER NUMBER
				2183

DATE MAILED: 04/05/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	10/074,779	DOWLING, ERIC M.	
	Examiner David J. Huisman	Art Unit 2183	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

- 1) Responsive to communication(s) filed on 25 January 2006.
- 2a) This action is FINAL. 2b) This action is non-final.
- 3) Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

- 4) Claim(s) 1-14, 16-32 and 34-50 is/are pending in the application.
- 4a) Of the above claim(s) _____ is/are withdrawn from consideration.
- 5) Claim(s) _____ is/are allowed.
- 6) Claim(s) 1-14, 16-32 and 34-50 is/are rejected.
- 7) Claim(s) _____ is/are objected to.
- 8) Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

- 9) The specification is objected to by the Examiner.
- 10) The drawing(s) filed on 16 May 2005 is/are: a) accepted or b) objected to by the Examiner.
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

- 12) Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) All b) Some * c) None of:
 1. Certified copies of the priority documents have been received.
 2. Certified copies of the priority documents have been received in Application No. _____.
 3. Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

- | | |
|---|---|
| 1) <input type="checkbox"/> Notice of References Cited (PTO-892) | 4) <input type="checkbox"/> Interview Summary (PTO-413) |
| 2) <input type="checkbox"/> Notice of Draftsperson's Patent Drawing Review (PTO-948) | Paper No(s)/Mail Date. _____. |
| 3) <input type="checkbox"/> Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08)
Paper No(s)/Mail Date _____. | 5) <input type="checkbox"/> Notice of Informal Patent Application (PTO-152) |
| | 6) <input type="checkbox"/> Other: _____. |

DETAILED ACTION

1. Claims 1-14, 16-32 and 34-50 have been examined.

Papers Submitted

2. It is hereby acknowledged that the following papers have been received and placed of record in the file: RCE and Amendment as received on 1/25/2006.

Claim Rejections - 35 USC § 103

3. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

4. Claims 1, 6-10, 12-14, 23-25, and 46-48 are rejected under 35 U.S.C. 103(a) as being unpatentable over Inagami et al., U.S. Patent No. 4,881,168 (herein referred to as Inagami). In addition, Wright et al., U.S. Patent No. 5,587,961 (herein referred to as Wright) is cited as extrinsic evidence for showing that precharge and deactivate commands exist.

5. Referring to claim 1, Inagami has taught a processor comprising:
 - a) an embedded array comprising a plurality of random access memory cells arranged in rows and columns. See Fig.1 and note that main memory (component 1) inherently comprises rows and columns of memory cells. Also, note that the memory is embedded in the overall system (Inagami states that Fig.1 shows a vector processor and the memory is part of the processor) Inagami has not explicitly taught that the memory array is a DRAM array. However, Official

Notice is taken that DRAM and its advantages are well known and expected in the art. More specifically, DRAM is a very popular memory technology because of its high density and low price (in comparison to other memory such as SRAM). Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagami's main storage to be a DRAM array.

b) a row address register that holds a pointer that points to a row of the DRAM array. See Fig.1, component 60, and column 5, lines 13-17.

c) one or more sets of registers, each of said sets of registers capable of being loaded or stored in response to a single latch signal. See Fig.1, components VMR0-VMR7 and VR0-VR7. Note that these registers may be loaded or stored in response to a single load or store signal shown in Fig.4, Fig.5, and column 1, lines 52-64.

d) an instruction set which includes:

(i) at least one command to perform arithmetic on said row address registers. See Fig.6, column 7, line 66, to column 8, line 9, and note that arithmetic is performed on the row address registers by the start address calculation unit 310.

(ii) a command to precharge (activate) the row pointed to by said row address register. This is deemed inherent by the examiner because in a DRAM, a row must be precharged before it may be read. See Wright, column 1, lines 43-59, for further information.

(iii) a command to deactivate said row pointed to by said row address register after it had been precharged by the command to precharge. This command is also deemed inherent because if the rows are not deactivated, they will stay activated. Clearly, after an access is made, that memory access is complete. After an access to a row, it must be precharged

so that it may be accessed again in the future. Therefore, it must be deactivated. See Wright, column 1, lines 37-59, for further information.

(iv) a command to load a plurality of words of the precharges into designated sets of data registers. See Fig.4, and column 1, lines 52-59.

(v) a command to load selected columns of the precharged row pointed to by said row address registers into designated sets of data registers, said selection based on bits in a mask. Again, see Fig.4, the abstract, and column 1, lines 52-59.

6. Referring to claim 6, Inagami has taught a processor as described in claim 1. Inagami has further taught a plurality of DRAM arrays. More specifically, the smallest array possible is either 1 row or 1 column. Clearly, Inagami has more rows and columns than this; otherwise, very little data would be stored. Consequently, Inagami would have many DRAM arrays (for instance, may 1x1 memory arrays).

7. Referring to claim 7, Inagami has taught a processor as described in claim 1. Inagami has further taught:

a) at least one functional unit. See Fig.1, component 5.
b) whereby said one or more sets of registers comprise a plurality of register files (see Fig.1, components Vro-VR7 and VMR0-VMR7), each of said register files comprising a parallel access port operative to load or store contents of said register file in a single cycle from or to a DRAM row as selected by said row-address register (again see Fig.1), each of said register files further comprising at least a second access port operative to transfer data between said functional unit and a selected subset register in said register file (note that the register files are coupled to the functional units 5 via switches).

8. Referring to claim 8, Inagami has taught a processor as described in claim 7. Inagami has further taught:

- a) a second functional unit. See Fig.1, component 2, for instance.
- b) whereby said first functional unit executes a first command to perform logical processing on the contents of one or more registers within a selected active one of said register sets (as seen in Fig.1, component 5 performs operations on register operands), and said second functional unit executes a second command to parallelly transfer data between a selected inactive one of said register sets and said DRAM array (note that the load/store pipes do the loading and that a register file may be loaded while not being used to supply operands, i.e., inactive).

9. Referring to claim 9, Inagami has taught a processor as described in claim 8. Inagami has not explicitly taught that said first and second functional units execute said first and second commands substantially contemporaneously. However, it should be noted that the load/store pipes and the operation pipes are completely separate (see Fig.1). That is, there is not a single unit which executes both instructions. Consequently, both the first and second commands could execute simultaneously and this would result in the most efficiency as two operations may be performed in less time than performing one after the other. As a result, it would have been obvious to execute the first and second commands simultaneously because Inagami's hardware supports it.

10. Referring to claim 10, Inagami has taught a processor as described in claim 8. Inagami has further taught:

- a) a first software module comprising a set of data manipulation commands, said first software module executed by said first functional unit. It is inherent that a group of instructions exist

which causes the manipulation of register data. These commands, which include multiplication, addition, etc. (column 4, lines 35-37), would be executed by the operation pipes 5 (Fig.1).

b) a second software module comprising a set of parallel data transfer commands, said second software module being executed by said second functional unit. See column 1, lines 52-64, and note that these transfer commands are executed by the second functional unit (load/store pipes). See column 4, lines 32-35.

c) whereby said second software module operates in support of said first software module to prefetch data from said DRAM array into one of said register files in advance of said data being needed by said first software module. Clearly, when load instructions are executed, they are executed to bring data into the memory so that subsequent instructions may use that data. This is prefetching in that the data is prefetched before the consumer instruction actually requires it.

11. Referring to claim 12, Inagami has taught a processor as described in claim 1. Inagami has further taught:

a) first and second sets of functional units (Fig.1, components 5 and 2 respectively), said first and second sets of functional units having respective first and second instruction subsets. Clearly, the first functional unit will execute instructions such as multiplication and addition (manipulation instructions) while the second functional unit executes instructions for data transfer. See column 4, lines 32-37.

b) whereby the second instruction subset includes said commands (ii), (iii), (iv) and (v). Since commands (ii)-(v) deal with memory, they will be executed by component 2.

c) Inagami has not explicitly taught that command (i) is part of the first instruction subset.

However, the row address register holds memory addresses and Official Notice is taken that

memory addresses may be generated by arithmetic units. This would allow for different types of addressing modes to exist, such as indirect addressing. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagami such that command (i) is part of the first instruction subset.

12. Referring to claim 13, Inagami has taught a processor comprising:
- a) an embedded array comprising a plurality of random access memory cells arranged in rows and columns. See Fig.1 and note that main memory (component 1) inherently comprises rows and columns of memory cells. Also, note that the memory is embedded in the overall system (Inagami states that Fig.1 shows a vector processor and the memory is part of the processor). Inagami has not explicitly taught that the memory array is a DRAM array. However, Official Notice is taken that DRAM and its advantages are well known and expected in the art. More specifically, DRAM is a very popular memory technology because of its high density and low price (in comparison to other memory such as SRAM). Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagami's main storage to be a DRAM array.
 - b) a row address register that holds a pointer that points to a row of the DRAM array. See Fig.1, component 60, and column 5, lines 13-17.
 - c) one or more sets of data registers, each of said sets of data registers capable of being loaded or stored in response to a single latch signal. See Fig.1, components VMR0-VMR7 and VR0-VR7. Note that these registers may be loaded or stored in response to a single load or store signal shown in Fig.4, Fig.5, and column 1, lines 52-64.

d) a bit mask to select one or more data locations within at least one of said register sets. See Fig.4 and Fig.5, component 22.

e) an instruction set which comprises at least:

(i) a command to perform arithmetic on said row address register. See Fig.6, column 7, line 66, to column 8, line 9, and note that arithmetic is performed on the row address registers by the start address calculation unit 310.

(ii) a command to precharge (activate) the row pointed to by said row address register. This is deemed inherent by the examiner because in a DRAM, a row must be precharged before it may be read. See Wright, column 1, lines 43-59, for further information.

(iii) a command to load a set of selected elements of the row pointed to by said row address register into a selected set of said data registers, said selection based on bits in said mask. See Fig.4, and column 1, lines 52-59.

f) wherein the command to precharge is executed to precharge the row prior to the command to load so that at the time the command to load is issued, the command to load can execute without the need to wait for the row to precharge. This is deemed inherent because before loading from a row can occur, the row must be precharged. And, see Wright, column 1, lines 43-59 and note that precharge and access commands are separate. So a precharge must occur before a load (so that the row is charged and as a result is able to be read).

13. Referring to claim 14, Inagami has taught a processor as described in claim 13. Inagami has further taught that said load command causes an entire row that was previously precharged to be loaded. Note from Fig.4 that if the mask were set to all 1's, then the entire row would be loaded.

14. Referring to claim 23, Inagami has taught a processor comprising:
- a) an embedded array comprising a plurality of random access memory cells arranged in rows and columns. See Fig.1 and note that main memory (component 1) inherently comprises rows and columns of memory cells. Also, note that the memory is embedded in the overall system (Inagami states that Fig.1 shows a vector processor and the memory is part of the processor). Inagami has not explicitly taught that the memory array is a DRAM array. However, Official Notice is taken that DRAM and its advantages are well known and expected in the art. More specifically, DRAM is a very popular memory technology because of its high density and low price (in comparison to other memory such as SRAM). Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagami's main storage to be a DRAM array.
 - b) a row address register that holds a pointer that points to a row of the DRAM array. See Fig.1, component 60, and column 5, lines 13-17.
 - c) first and second register files, each of said register files capable of being loaded or stored in response to a single latch signal. See Fig.1, components VMR0-VMR7 and VR0-VR7. Note that these registers may be loaded or stored in response to a single load or store signal shown in Fig.4, Fig.5, and column 1, lines 52-64.
 - d) an embedded processor comprising first and second functional units, said first and second functional units having respective first and second instruction sets and capable of accessing said first and second register files. See Fig.1 and note that the first set of functional units could be component 5, which executes manipulation instructions like multiplication, addition, etc. See

column 4, lines 35-37. The second set of functional units could be component 2 of Fig.1, which executes data transfer instructions such as loads and stores. See column 4, lines 32-35.

e) wherein said first and second register files comprise a parallel access port (see connections between register files and switches 120-123) operative to parallelly transfer contents of one of said register files between a DRAM row as selected by said row-address register (again see Fig.1 and note that these connections allow register files to be loaded with DRAM contents), said first and second register files further comprising at least a second access port operative to transfer data between a selected one of said register files and said second functional unit (see the connections in Fig.1 between the switches 120-123 and the load/store pipes (second functional unit...these allow data to be read from the register files by the pipes and ultimately stored in DRAM).

f) wherein said first instruction set comprises at least:

(i) a command to manipulate data in a data register with a register file. See column 4, lines 35-37.

g) wherein said second instruction set comprises at least:

(i) a command to perform arithmetic on said row address register. See Fig.6, column 7, line 66, to column 8, line 9, and note that arithmetic is performed on the row address registers by the start address calculation unit 310.

15. (ii) a command to load the entire row pointed to by said row address register into a selected set of registers of said register files. See Fig.4, and column 1, lines 52-59. Note that each row of the memory shown in Fig.4 which corresponds to a mask bit equal to 1 is transferred

to a register file. For instance, a0, which is clearly the first row in the main storage, is transferred to a register file. The data a0 is the entire row data.

16. Referring to claim 24, Inagami has taught a processor as described in claim 23. Inagami has not explicitly taught that said first and second functional units each respectively execute a command from said first and second instruction sets substantially contemporaneously. However, it should be noted that the load/store pipes 2 and the operation pipes 5 are completely separate (see Fig.1). That is, there is not a single unit which executes both instructions. Consequently, both the first and second commands (for instance, an add and a load) could execute simultaneously and this would result in the most efficiency as two operations may be performed in less time than performing one after the other. As a result, it would have been obvious to execute the first and second commands simultaneously because Inagami's hardware supports it.

17. Referring to claim 25, Inagami has taught a processor as described in claim 24. Inagami has further taught:

a) a first software module comprising data manipulation commands drawn from said first instruction set, said first software module executed by said first functional unit. It is inherent that a group of instructions exist which causes the manipulation of register data. These commands, which include multiplication, addition, etc. (column 4, lines 35-37), would be executed by the operation pipes 5 (Fig.1).

b) a second software module comprising a parallel data transfer command drawn from said second instruction set, said second software module being executed by said second functional unit. See column 1, lines 52-64, and note that these transfer commands are executed by the second functional unit (load/store pipes). See column 4, lines 32-35.

c) whereby said second software module operates in support of said first software module to prefetch data from said DRAM array into one of said register files in advance of said data being needed by said first software module. Clearly, when load instructions are executed, they are executed to bring data into the memory so that subsequent instructions may use that data. This is prefetching in that the data is prefetched before the consumer instruction actually requires it.

18. Referring to claim 46, Inagami has taught a digital processor comprising:

a) an embedded array having a plurality of random access memory cells arranged in rows and columns. See Fig.1 and note that main memory (component 1) inherently comprises rows and columns of memory cells. Also, note that the memory is embedded in the overall system (Inagami states that Fig.1 shows a vector processor and the memory is part of the processor). Inagami has not explicitly taught that the memory array is a DRAM array. However, Official Notice is taken that DRAM and its advantages are well known and expected in the art. More specifically, DRAM is a very popular memory technology because of its high density and low price (in comparison to other memory such as SRAM). Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagami's main storage to be a DRAM array.

b) at least one row address register. See Fig.1, component 60, and column 5, lines 13-17.
c) one or more sets of registers capable of being loaded or stored. See Fig.1, components VMR0-VMR7 and VR0-VR7. Note that these registers may be loaded or stored in response to a single load or store signal shown in Fig.4, Fig.5, and column 1, lines 52-64.
d) a method of processing data comprising:

(i) performing arithmetic on said at least one row address register in order to manipulate a pointer that points to a row of the embedded DRAM array. See Fig.6, column 7, line 66, to column 8, line 9, and note that arithmetic is performed on the row address registers by the start address calculation unit 310.

(ii) precharging (activating) a row pointed to by said at least one row address register. This is deemed inherent by the examiner because in a DRAM, a row must be precharged before it may be read. See Wright, column 1, lines 43-59, for further information.

Inagami has not taught speculatively precharging based upon a possible anticipated need to perform one or more load and or store operations that would access said row.

However, Official Notice is taken that prefetching is a well known and advantageous concept in the art. Prefetching allows data to be fetched before it is actually needed.

Therefore, when it is needed, it will be available immediately, as opposed to fetching it at that time. This will increase efficiency. Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagami to include prefetching. Just from the concept of prefetching alone, it should be realized that it equates to speculative fetching. That is you speculate that the data will be needed in the near future and so the system fetches it ahead of time to increase efficiency. Whether a normal fetch or speculative fetch occurs, precharging still has to occur since a DRAM is being read. So, in a normal fetch, the row is precharged, and in a speculative fetch (prefetch), the row is speculatively precharged.

(iii) in response to a separate command executed after the speculatively precharging, loading a plurality of words of a row designated by said at least one row address register

into designated sets of data registers. See Fig.4, and column 1, lines 52-59. Also, Wright shows that the precharge and access commands are separate. See column 1, lines 43-59 (lines 43-45, specifically).

19. Referring to claim 47, Inagami has taught a method as described in claim 46. Inagami has further taught deactivating rows pointed to by said at least one row address register. This command is deemed inherent as if the rows are not deactivated, they will stay activated. Clearly, after an access is made, that memory access is complete. After an access to a row, it must be precharged so that it may be accessed again in the future. Therefore, it must be deactivated. See Wright, column 1, lines 37-59, for further information.

20. Referring to claim 48, Inagami has taught a digital processor comprising:

- a) an embedded array having a plurality of random access memory cells arranged in rows and columns. See Fig.1 and note that main memory (component 1) inherently comprises rows and columns of memory cells. Also, note that the memory is embedded in the overall system (Inagami states that Fig.1 shows a vector processor and the memory is part of the processor). Inagami has not explicitly taught that the memory array is a DRAM array. However, Official Notice is taken that DRAM and its advantages are well known and expected in the art. More specifically, DRAM is a very popular memory technology because of its high density and low price (in comparison to other memory such as SRAM). Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagami's main storage to be a DRAM array.
- b) a set of row address registers. See Fig.1, component 60, and column 5, lines 13-17.

c) one or more sets of registers each capable of being loaded or stored in response to a latch signal. See Fig.1, components VMR0-VMR7 and VR0-VR7. Note that these registers may be loaded or stored in response to a single load or store signal shown in Fig.4, Fig.5, and column 1, lines 52-64.

d) a method of processing data comprising:

(i) performing arithmetic on said row address registers. See Fig.6, column 7, line 66, to column 8, line 9, and note that arithmetic is performed on the row address registers by the start address calculation unit 310.

(ii) precharging (activating) rows pointed to by said row address registers. This is deemed inherent by the examiner because in a DRAM, a row must be precharged before it may be read. See Wright, column 1, lines 43-59, for further information.

(iii) in response to an instruction issued after the precharging, loading selected columns of rows of said embedded DRAM array pointed to by said row address registers into designated sets of said data registers, said selection based on bits in a mask. See Fig.4, the abstract, and column 1, lines 52-59. Also, note from Wright, column 1, lines 43-59 (lines 43-45, specifically), that separate commands for precharging and accessing are required.

21. Claims 2-5, 11, and 26-27 are rejected under 35 U.S.C. 103(a) as being unpatentable over Inagami, as applied above, in view of Parady, U.S. Patent No. 5,933,627.

22. Referring to claim 2, Inagami has taught a processor as described in claim 1.

- a) Inagami has further taught first and second sets of functional units, said first and second sets of functional units having respective first and second instruction sets and capable of accessing first and second sets of said register. See Fig.1, components 5-0 to 5-3, and note that these functional units may access any of the register sets.
- b) Inagami has not taught a command to select one of said first and second sets of registers to be an architectural set of registers accessible to said first set of functional units, a command to deselect the other of said first and second sets of registers so that it is no longer an architectural register set accessible to said first set of functional units, a command to select one of said first and second sets of registers to be an architectural set of registers accessible to said second set of functional units, and a command to deselect the other one of said first and second sets of registers so that it is no longer an architectural register set accessible to said second set of functional units. However, Parady has taught a thread switch command which performs such operations. More specifically, in Parady, a first set of functional units may comprise components 38 and 40 in Fig.1 (which would execute a first set of instructions comprising floating-point addition, subtraction, and multiplication operations). In response to a thread switch command, a new register set (Fig.1, component 50) corresponding to the switched-in (current) thread will be made accessible to the first set of functional units. Meanwhile, the previously used register set corresponding to the switch-out (inactive) thread is no longer accessed by the first set of functional units. See the abstract and column 2, lines 18-39. Likewise, the thread switch command will also make the new register set available to a second set of functional units (for example, components 42 and 44 in Fig.1, which would execute a second set of instructions comprising floating-point division and graphical addition and subtraction operations), whereas

the previously used register set will become “invisible” to the second set of functional units. A person of ordinary skill in the art would have recognized that multithreading and thread switching are beneficial tools for a system because when one thread stalls, another thread may be switched in and executed, thereby preventing the processor from going idle. With thread switching comes the switching of register files as well. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagami to toggle between active and inactive register file states because switching of register sets supports thread switching, which as described above, prevents a processor from staying idle during a stall, thereby increasing throughput and efficiency.

23. Referring to claim 3, Inagami has taught a processor as described in claim 1.

a) Inagami has further taught first and second sets of functional units, said first and second sets of functional units having respective first and second instruction sets and capable of accessing first and second sets of said registers. See Fig.1, components 5-0 to 5-3, and note that these functional units may access any of the register sets.

b) Inagami has not taught a command which selects one of said first and second sets of registers to be an architectural set of registers accessible to said first set of functional units, and at the same time, which deselects the other one of said one of said first and second sets of registers to be an architectural set of registers accessible to said second set of functional units. However, Parady has taught a thread switch command which performs such operations. More specifically, in Parady, a first set of functional units may comprise components 34 and 36 in Fig.1 (which would execute a first set of instructions comprising integer ALU, multiplication, and division operations). In response to a thread switch command, a new register set (Fig.1, component 48)

corresponding to the switched-in (current) thread will be made accessible to the first set of functional units (integer registers for integer functional units). At the same time, the new register set will not be accessible to a second set of functional unit comprising components 38 and 40 in Fig.1 (which would execute floating-point addition, subtraction, and multiplication operations). This is because the selected register file is an integer register file and floating-point functional units would not access the integer register file (they would access a floating-point register file 50). A person of ordinary skill in the art would have recognized that multithreading and thread switching are beneficial tools for a system because when one thread stalls, another thread may be switched in and executed, thereby preventing the processor from going idle. With thread switching comes the switching of register files as well. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagami to toggle between active and inaccessible register file states because switching of register sets supports thread switching, which as described above, prevents a processor from staying idle during a stall, thereby increasing throughput and efficiency.

24. Referring to claim 4, Inagami has taught a processor as described in claim 1. Inagami has not explicitly taught the specifics of the functional units 5-0 to 5-3, and more specifically has not taught first and second sets of functional units, said first and second sets of functional units having respective first and second instruction sets and accessing first and second sets of said registers and whereby said first and second instruction sets are subsets of said instruction set of said embedded-DRAM processor. However, Parady has taught the functional units may be of a floating-point type and of an integer type. See Fig.1. These two sets of functional units would execute integer and floating-point instructions, respectively, wherein each is a subset of the

overall instruction set (integer + floating-point + miscellaneous instructions). Having different types of functional unit allows for the execution of different types of instructions and consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagami to include different types of functional units.

25. Referring to claim 5, Inagami in view of Parady has taught a processor as described in claim 4. Inagami has not explicitly taught that said second set of functional units comprises a functional unit that is a multi-issue functional unit and further comprises a dispatch unit and a plurality of functional units which each execute a respective instruction stream as dispatched by said dispatch unit. However, Parady has taught such a concept. Note that the second functional unit may comprise components 28, 38, 40, 42, 44, and 46 of Fig.1. This includes a dispatch unit, which must inherently exist to dispatch instructions (in this case 4 instructions at a time (column 3, lines 14-17)), and a plurality of functional units (38-46) which may each execute dispatched instructions. Having multiple functional units and a multi-issue dispatcher allows multiple instructions to be executed at any given time, thereby increasing throughput. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagami to have a multi-issue functional unit.

26. Referring to claim 11, Inagami has taught a processor as described in claim 10. Inagami has further taught that:

a) said first software module contains an instruction that reference registers within an architectural register set visible to said first functional unit, whereby said architectural register set corresponds to at least partially to said one of said register files that is in an active state. See column 4, lines 35-37, and note addition and multiplication instructions would exist which

operate on data retrieved from register files (Fig.1). Clearly, if the functional unit 5 is retrieving data from a register file, then that register file is visible and in an active state.

b) said second software module contains instructions that cause data to be transferred between an inactive register set and said DRAM array. As discussed above, a register file does not need to be read every cycle. Therefore, it may be inactive with respect to the functional units. However, since the load pipes are separate from the functional units, the data transfer may occur even if the file is not being used by a functional unit.

c) Inagami has not taught that the second software module also executes a command to toggle a selected register set between said active and inactive states. However, Parady has taught a thread switch command which performs such toggling. More specifically, in Parady, every time a thread switch occurs, a new register file becomes active and the previously active register file becomes inactive. This allows for thread switching without having to save or reload a context of a thread, thereby saving processing time. A person of ordinary skill in the art would have recognized that multithreading and thread switching are beneficial tools for a system because when one thread stalls, another thread may be switched in and executed, thereby preventing the processor from going idle. With thread switching comes the switching of register files as well. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagami to toggle between active and inactive register file states because switching of register sets supports thread switching, which as described above, prevents a processor from staying idle during a stall, thereby increasing throughput and efficiency.

27. Referring to claim 26, Inagami has taught a processor as described in claim 23. Inagami has not taught that the second instruction set further comprises a command to toggle a register set

between an active state and an inactive state. However, Parady has taught a thread switch command which performs such toggling. More specifically, in Parady, every time a thread switch occurs, a new register file becomes active and the previously active register file becomes inactive. This allows for thread switching without having to save or reload a context of a thread, thereby saving processing time. A person of ordinary skill in the art would have recognized that multithreading and thread switching are beneficial tools for a system because when one thread stalls, another thread may be switched in and executed, thereby preventing the processor from going idle. With thread switching comes the switching of register files as well. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagami to toggle between active and inactive register file states because switching of register sets supports thread switching, which as described above, prevents a processor from staying idle during a stall, thereby increasing throughput and efficiency.

28. Referring to claim 27, Inagami in view of Parady has taught a processor as described in claim 26. Parady has further taught that said toggle command causes said first register file to toggle from the inactive state to the active state and also causes the second register file to toggle from the active state to the inactive state. Again, as described in the rejection of claim 23 above, every time a thread switch occurs, a new register file becomes active and the previously active register file becomes inactive. That is, each thread has its own register file (column 2, lines 35-37). Consequently, when a thread switch occurs, the register file corresponding to the switched in thread goes from inactive to active, while the register file corresponding to the switched out thread goes from active to inactive.

29. Claims 16-22, 28-32 and 34-45 and 49-50 are rejected under 35 U.S.C. 103(a) as being unpatentable over Inagami in view of Parady, as applied above, and further in view of Bissett et al., U.S. Patent No. 5,896,523 (herein referred to as Bissett).

30. Referring to claim 16, Inagami has taught a processor comprising:
- a) an embedded array comprising a plurality of random access memory cells arranged in rows and columns. See Fig.1 and note that main memory (component 1) inherently comprises rows and columns of memory cells. Also, note that the memory is embedded in the overall system (Inagami states that Fig.1 shows a vector processor and the memory is part of the processor). Inagami has not explicitly taught that the memory array is a DRAM array. However, Official Notice is taken that DRAM and its advantages are well known and expected in the art. More specifically, DRAM is a very popular memory technology because of its high density and low price (in comparison to other memory such as SRAM). Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagami's main storage to be a DRAM array.
 - b) a row address register that holds a pointer that points to a row of the DRAM array. See Fig.1, component 60, and column 5, lines 13-17.
 - c) first and second register files, each of said register files having a plurality of data registers capable of being loaded or stored in response to a single latch signal. See Fig.1, components VMR0-VMR7 and VR0-VR7. Note that these registers may be loaded or stored in response to a single load or store signal shown in Fig.4, Fig.5, and column 1, lines 52-64.
 - d) Inagami has not taught that each of said register files is also capable of being placed into an active state and an inactive state. However, Parady has taught a thread switch command which

performs such state toggling. More specifically, in Parady, every time a thread switch occurs, a new register file becomes active and the previously active register file becomes inactive. This allows for thread switching without having to save or reload a context of a thread, thereby saving processing time. A person of ordinary skill in the art would have recognized that multithreading and thread switching are beneficial tools for a system because when one thread stalls, another thread may be switched in and executed, thereby preventing the processor from going idle. With thread switching comes the switching of register files as well. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagami to toggle between active and inactive register file states because switching of register sets supports thread switching, which as described above, prevents a processor from staying idle during a stall, thereby increasing throughput and efficiency.

- e) a set of functional units that perform logical operations on data accessed from a set of architectural registers, wherein registers placed into the active state appear as architectural registers to a set of functional units, and registers in the inactive state are not accessible by the functional units. See Fig.1 and Fig.3 of Parady. Note that multiple functional units (36, 38, 40, etc) would operate on instructions from the current thread, which has an associated active register file (Fig.3, files 48-50). Therefore, the active file is available to the functional units while the files for the inactive threads are not available to the units.
- f) a bit mask to select one or more locations within at least one of said register files. See Fig.4 and Fig.5, component 22.
- g) an instruction set which comprises at least:

(i) a command to perform arithmetic on said row address register. See Fig.6, column 7, line 66, to column 8, line 9, and note that arithmetic is performed on the row address registers by the start address calculation unit 310.

(ii) a command to load a set of selected elements of the row pointed to by said row address register into a selected set of said data registers, said selection based on bits in said mask. See Fig.4, and column 1, lines 52-59. Inagami in view of Parady has not taught that the selected set of said data registers to be loaded are in the inactive state. However, Bissett has taught performing data prefetches (loads) in the background. And, this is beneficial because background operations do not influence software execution, thereby allowing the current program to run normally while also accomplishing a task in the background. See column 4, lines 3-8. And, as is known in the art, prefetching is beneficial because data is brought in from memory before an instruction needs it. Then, when the instruction does actually execute, the data is already fetched, allowing the instruction to execute more quickly. Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagami in view of Parady in view of Bissett such that Inagami in view of Parady performs background prefetching to a register file is not currently in use (one that corresponds to an inactive thread).

31. Referring to claim 17, Inagami in view of Parady has taught a processor as described in claim 16. Parady has further taught that the instruction set further comprises a command to toggle a register set between said active and inactive states. As discussed in Parady's abstract, a thread switch occurs on a cache miss during a load instruction's execution. Therefore, this command is built into a load instruction which is part of the instruction set.

32. Referring to claim 18, Inagami in view of Parady has taught a processor as described in claim 17. Parady has further taught that said toggle command causes said first register file to toggle from the inactive state to the active state and also causes the second register file to toggle from the active state to the inactive state. Again, as described in the rejection of claim 16 above, every time a thread switch occurs, a new register file becomes active and the previously active register file becomes inactive. That is, each thread has its own register file (column 2, lines 35-37). Consequently, when a thread switch occurs, the register file corresponding to the switched in thread goes from inactive to active, while the register file corresponding to the switched out thread goes from active to inactive.

33. Referring to claim 19, Inagami in view of Parady has taught a processor as described in claim 16. Inagami has further taught that the instruction set further comprises a command to manipulate the bits in the bit mask. Although the command is not explicitly mentioned, the vector masks are stored in register files VMR0-VMR7 and they may be written to (modified) as seen in Fig.1.

34. Referring to claim 20, Inagami in view of Parady has taught a processor as described in claim 16. Inagami has further taught:

a) first and second sets of functional units, said first and second sets of functional units having respective first and second sets of instructions and capable of accessing said first and second register sets. See Fig.1 and note that the first set of functional units could be component 5, which executes manipulation instructions like multiplication, addition, etc. See column 4, lines 35-37. The second set of functional units could be component 2 of Fig.1, which executes data transfer instructions such as loads and stores. See column 4, lines 32-35.

b) said instruction set further comprises at least:

- (i) a command to select one of said first and second sets of registers to be an architectural set of registers accessible to said first set of functional units. See Fig.1. Clearly, if a functional unit (5-0, for instance) executes an addition instruction, it will select at least one of the register files so that it may access operands.
- (ii) a command to select one of said first and second sets of registers to be an architectural set of registers accessible to said second set of functional units. See Fig.1. Clearly, if a functional unit (2-0, for instance) executes a store instruction, it will select at least one of the register files so that it may access an operand to be stored.

35. Referring to claim 21, Inagami in view of Parady has taught a processor as described in claim 20. Inagami has not taught that the instruction set further comprises a command to deselect the other of said first and second sets of registers so that it is no longer an architectural register set accessible to said first set of functional units and a command to deselect the other one of said first and second sets of registers so that it is no longer an architectural register set accessible to said second set of functional units. However, Parady has taught such a concept. More specifically, in Parady, every time a thread switch occurs, a new register file becomes active and the previously active register file becomes inactive (deselected so it is no longer accessible). This allows for thread switching without having to save or reload a context of a thread, thereby saving processing time. A person of ordinary skill in the art would have recognized that multithreading and thread switching are beneficial tools for a system because when one thread stalls, another thread may be switched in and executed, thereby preventing the processor from going idle. With thread switching comes the switching of register files as well.

Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagami to toggle between selection and deselection of register files because switching of register sets supports thread switching, which as described above, prevents a processor from staying idle during a stall, thereby increasing throughput and efficiency.

36. Referring to claim 22, Inagami in view of Parady has taught a processor as described in claim 20. Inagami has further taught that at least one of said sets of functional units contains a single functional unit. See Fig.1, component 5, and note that it contains a single unit 5-0.

37. Referring to claim 28, Inagami has taught a processor comprising:

a) an embedded array comprising a plurality of random access memory cells arranged in rows and columns. See Fig.1 and note that main memory (component 1) inherently comprises rows and columns of memory cells. Also, note that the memory is embedded in the overall system (Inagami states that Fig.1 shows a vector processor and the memory is part of the processor). Inagami has not explicitly taught that the memory array is a DRAM array. However, Official Notice is taken that DRAM and its advantages are well known and expected in the art. More specifically, DRAM is a very popular memory technology because of its high density and low price (in comparison to other memory such as SRAM). Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagami's main storage to be a DRAM array.

b) first and second dual-port register files, each of said register files capable of parallelly transferring data between a row of said DRAM array. See Fig.1, components VMR0-VMR7 and VR0-VR7. Note that these registers may be loaded or stored in response to a single load or store

signal shown in Fig.4, Fig.5, and column 1, lines 52-64. Also, it should be noted that these files are at least dual-port in that they have ports for reading and writing (see Fig.1)

c) Inagami has not taught that each of said register files is also capable of being placed into an active state and an inactive state. However, Parady has taught a thread switch command which performs such state toggling. More specifically, in Parady, every time a thread switch occurs, a new register file becomes active and the previously active register file becomes inactive. This allows for thread switching without having to save or reload a context of a thread, thereby saving processing time. A person of ordinary skill in the art would have recognized that multithreading and thread switching are beneficial tools for a system because when one thread stalls, another thread may be switched in and executed, thereby preventing the processor from going idle. With thread switching comes the switching of register files as well. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagami to toggle between active and inactive register file states because switching of register sets supports thread switching, which as described above, prevents a processor from staying idle during a stall, thereby increasing throughput and efficiency.

d) first and second embedded functional units, said first and second functional units having respective first and second instruction sets that operate on registers of an architectural register set, said architectural register set comprising one of the first and second dual-port register files that is currently in the active state. See Fig.1 and note that the first set of functional units would be component 5, which executes manipulation instructions like multiplication, addition, etc. See column 4, lines 35-37. The second set of functional units would be component 2 of Fig.1, which

executes data transfer instructions such as loads and stores. See column 4, lines 32-35. Also, as taught in Parady, only the register files in the active state would be accessible.

e) whereby said first instruction set comprises at least:

(i) a command to manipulate data in a data register with a register file. See column 4, lines 35-37.

f) whereby said second instruction set comprises at least:

(i) a command to place an said selected inactive data register file into said active state, wherein when the inactive register file is activated, it becomes an architectural register set of said first functional unit. See Parady, and note the thread switch command that from the abstract and from Fig.3. More specifically, when a thread switch occurs, the register file associated with the “switched-in” thread will become active and consequently used by the system during execution of that thread.

(ii) Inagami has taught a command to unidirectionally transfer data between a row of said DRAM array and a data register file. See Fig.4. Neither Inagami nor Parady have taught such a transfer involves an inactive register file. However, Bissett has taught performing data prefetches (loads) in the background. And, this is beneficial because background operations do not influence software execution, thereby allowing the current program to run normally while also accomplishing a task in the background. See column 4, lines 3-8. And, as is known in the art, prefetching is beneficial because data is brought in from memory before an instruction needs it. Then, when the instruction does actually execute, the data is already fetched, allowing the instruction to execute more quickly.

Consequently, it would have been obvious to one of ordinary skill in the art at the time of

the invention to modify Inagami in view of Parady in view of Bissett such that Inagami in view of Parady performs background prefetching to a register file is not currently in use (one that corresponds to an inactive thread).

(iii) ***NOTE*** As an alternate interpretation, an inactive register file could be nothing more than a file which is not currently being accessed by the functional units. This could be the case in Fig.1, where there are many register files. Some may not be needed by functional units 5. These would be inactive and a load instruction may be the instructions used to transfer data to the inactive file. In addition, when the functional units do request operands from the inactive file (say in response to an add instruction), then the inactive file becomes active and accessible to the functional unit. Applicant should amend accordingly to avoid both interpretations of Inagami, etc.

38. Referring to claim 29, Inagami in view of Parady and further in view of Bissett has taught a processor as described in claim 28. Inagami has further taught that said command to unidirectionally transfer data causes data to be transferred from a row of the DRAM array to said selected inactive data register file. See Fig.4.

39. Referring to claim 30, Inagami in view of Parady and further in view of Bissett has taught a processor as described in claim 28. Inagami has further taught that said command to unidirectionally transfer data causes data to be transferred from said selected inactive data register file to a row of the DRAM array. See Fig.5.

40. Referring to claim 31, Inagami in view of Parady and further in view of Bissett has taught a processor as described in claim 28. Parady has further taught that the said command to place the selected inactive register file into the active state is a command that also causes the remaining

register file to toggle from the active state into the inactive state. As discussed in Parady's abstract, a thread switch occurs on a cache miss during a load instruction's execution. Therefore, this command is built into a load instruction which is part of the instruction set.

41. Referring to claim 32, Inagami in view of Parady and further in view of Bissett has taught a processor as described in claim 28.

a) Inagami has further taught at least one additional register file. Note that there are at least 16 register files shown in Fig.1 (VMR0-VMR7 and VR0-VR7).

b) Parady has further taught that said command to place the selected inactive register file into the active state is a command that also causes a selected other register file to toggle from the active state into the inactive state. As discussed in Parady's abstract, a thread switch occurs on a cache miss during a load instruction's execution. Therefore, this command is built into a load instruction which is part of the instruction set.

42. Referring to claim 34, Inagami in view of Parady and further in view of Bissett has taught a processor as described in claim 28. Inagami has further taught:

a) at least one bit mask. See Fig.4, component 22.

b) the second instruction set further comprises a command to move a subset of elements between a selected register file and a selected row of said DRAM array, whereby said subset is identified by said bit mask. See fig.4 and Fig.5.

43. Referring to claim 35, Inagami in view of Parady and further in view of Bissett has taught a processor as described in claim 28.

a) Inagami has not explicitly taught a dispatch unit but it is deemed inherent by the examiner that such a component exists. That is, instructions must be dispatched somehow to the functional units. Consequently, a dispatch unit is required.

b) Inagami has further taught a plurality of functional units that each execute a respective instruction stream as dispatched by said dispatch unit. See functional units 5-0 to 5-3 in Fig.1. Each of units 5 may execute mult, add, etc., all of which must be dispatched.

c) Inagami has not explicitly taught that the first functional unit is a multi-issue functional unit. However, Parady has taught such a concept. Note that the functional unit of Parady may comprise components 28, 38, 40, 42, 44, and 46 of Fig.1. This includes a dispatch unit, which must inherently exist to dispatch instructions (in this case 4 instructions at a time (column 3, lines 14-17)), and a plurality of functional units (38-46) which may each execute dispatched instructions. Having multiple functional units and a multi-issue dispatcher allows multiple instructions to be executed at any given time, thereby increasing throughput. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagami to have a multi-issue functional unit.

44. Referring to claim 36, Inagami in view of Parady and further in view of Bissett has taught a processor as described in claim 28. Inagami has further taught:

a) a first software module comprising a set of data manipulation commands drawn from said first instruction set, said first software module executed by said first functional unit. It is inherent that a group of instructions exist which causes the manipulation of register data. These commands, which include multiplication, addition, etc. (column 4, lines 35-37), would be executed by the operation pipes 5 (Fig.1).

b) a second software module comprising a set of parallel data transfer commands drawn from said second instruction set, said second software module being executed by said second functional unit. See column 1, lines 52-64, and note that these transfer commands are executed by the second functional unit (load/store pipes). See column 4, lines 32-35.

c) whereby said second software module operates in support of said first software module to prefetch data from said DRAM array into one of said register files in advance of said data being needed by said first software module. Clearly, when load instructions are executed, they are executed to bring data into the memory so that subsequent instructions may use that data. This is prefetching in that the data is prefetched before the consumer instruction actually requires it.

45. Referring to claim 37, Inagami in view of Parady and further in view of Bissett has taught a processor as described in claim 28. Inagami has further taught that:

a) said first software module contains an instruction that references registers within an architectural register set visible to said first functional unit, whereby said architectural register set corresponds to at least partially to said one of said register files that is in an active state. See column 4, lines 35-37, and note addition and multiplication instructions would exist which operate on data retrieved from register files (Fig.1). Clearly, if the functional unit 5 is retrieving data from a register file, then that register file is visible and in an active state.

b) said second software module contains instructions that cause data to be transferred between an inactive register set and said DRAM array. As discussed above, a register file does not need to be read every cycle. Therefore, it may be inactive with respect to the functional units. However, since the load pipes are separate from the functional units, the data transfer may occur even if the file is not being used by a functional unit.

c) Finally, recall that Parady has taught that the second software module also executes a command to toggle a selected register set between said active and inactive states. More specifically, Parady has taught a thread switch command which performs such toggling. Every time a thread switch occurs, a new register file becomes active and the previously active register file becomes inactive.

46. Referring to claim 38, Inagami in view of Parady and further in view of Bissett has taught a processor as described in claim 28. Inagami has further taught that each of said register files contain a number of words, N, matched to the number of words in a row of said DRAM array, and said unidirectional transfer comprises moving said selected row in its entirety to said selected register file. See Fig.4 and note that if the vector is all 1's then all of the rows will be moved to all of the registers. That is the register file may accommodate N words from DRAM.

47. Referring to claim 39, Inagami in view of Parady and further in view of Bissett has taught a processor as described in claim 28. Inagami has further taught a mask and switch unit interposed between said DRAM array and at least one of said register files. See Fig.4 and Fig.5.

48. Referring to claim 40, Inagami in view of Parady and further in view of Bissett has taught a processor as described in claim 28. Inagami in view of Parady and further in view of Bissett has not explicitly taught that said second set of instructions comprises a command to cause data to be moved from one register to another within a given one of said register files (individual register-to-register move operations). However, Official Notice is taken the register-register move operations are well known and expected in the art. These operations at the very least allow for the copying of registers. Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to implement register-register move operations.

49. Referring to claim 41, Inagami in view of Parady and further in view of Bissett has taught a processor as described in claim 28. Inagami has further taught that said second instruction set is used to implement an intelligent caching scheme, whereby said register files act as a cache and said second set of instructions are executed in lieu of a standard cache that maintains most recently used data and enforces a set associative or a direct-mapped caching policy: It should be noted that caches are not discussed in any way whatsoever in Inagami. Consequently, it is determined that Inagami does not employ a cache. And, register files hold recent operand data which is to be used by the processor for operations. As a result, the register files act as a cache for holding recent operation data.

50. Referring to claim 42, Inagami in view of Parady and further in view of Bissett has taught a processor as described in claim 28.

a) Inagami has not explicitly taught an instruction register coupled to receive instructions from said instruction set, said instruction register operative to hold an instruction to be executed by a data assembly unit. However, the examiner has deemed an instruction register (IR) as being a component which inherently exists within a system that executes instructions. That is, when instructions are fetched from memory, they must be held in an IR so that they may be decoded and executed. This includes instructions to be executed by the data assembly unit (Fig.1, component 2).

b) Inagami has not explicitly taught a local program memory, but this is also inherent as instructions must be stored in some form of memory if they are to be executed.

c) Inagami has not taught that said second functional unit corresponds to said data assembly unit, and said data assembly unit receives an instruction from said second instruction set that causes a

separate control thread of instructions to be accessed from said local program memory and executed by said data assembly unit. However, Parady has taught this concept. As discussed above, when a load is executed (by the data assembly unit) but it misses the cache, a thread switch occurs, thereby causing a new thread of instructions to be accessed and executed.

51. Referring to claim 43, Inagami in view of Parady and further in view of Bissett has taught a processor as described in claim 42.

a) Inagami has not explicitly taught a prefetch unit that prefetches instructions from the first and second instruction sets from a single very long instruction word (VLIW) instruction memory. However, Official Notice is taken that both prefetching and VLIW instructions are well known concepts in the art. Prefetching allows instructions/data to be fetched before they are actually needed. Therefore, when they are needed, they will be available immediately, as opposed to fetching them at that time. This will increase efficiency. Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagami to include a prefetch unit. Also, VLIW instructions allow for multiple instructions to be executed at the same time and since they are bundled together during compilation, the need for dynamic scheduling is reduced. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagami to implement VLIW instructions.

b) Inagami has not explicitly taught a dispatch unit that dispatches instructions from the first instruction set to the functional units and dispatches instructions from the second instruction stream to the data assembly unit. However, such a component is deemed as inherently existing by the examiner. That is, instructions must be dispatched somehow to the functional units. Consequently, a dispatch unit is required.

52. Referring to claim 44, Inagami in view of Parady and further in view of Bissett has taught a processor as described in claim 28.

- a) Inagami has further taught a command to precharge a row of the DRAM array. This is deemed inherent by the examiner because in a DRAM, a row must be precharged before it may be read. See Wright, column 1, lines 43-59, for further information.
- b) Also recall that it would have been obvious to modify Inagami to perform prefetching. When prefetching, the system must monitor what instructions it is prefetching for, otherwise it will not know what data to prefetch. In addition, when prefetching, the unit would perform speculative precharging as the DRAM row needs to be precharged in order to fetch from it. It would be speculative because the precharging is done before it is known whether the prefetched data will actually be used or not.

53. Referring to claim 45, Inagami has taught a processor comprising:

- a) an embedded array comprising a plurality of random access memory cells. See Fig.1 and note that main memory (component 1) inherently comprises memory cells. Also, note that the memory is embedded in the overall system (Inagami states that Fig.1 shows a vector processor and the memory is part of the processor). Inagami has not explicitly taught that the memory array is a DRAM array. However, Official Notice is taken that DRAM and its advantages are well known and expected in the art. More specifically, DRAM is a very popular memory technology because of its high density and low price (in comparison to other memory such as SRAM). Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagami's main storage to be a DRAM array.

- b) first and second dual-port register files (see Fig.1, components VMR0-VMR7 and VR0-VR7), whereby the first port of each of said register files is a parallel access port and is parallelly coupled to said DRAM array (see Fig.1, note that if transfers occur between DRAM and registers, as shown in Fig.4 and Fig.5, then a port must couple registers to DRAM).
- c) Inagami has not taught that each of said register files is capable of being placed into an active state and an inactive state. However, Parady has taught a thread switch command which performs such state toggling. More specifically, in Parady, every time a thread switch occurs, a new register file becomes active and the previously active register file becomes inactive. This allows for thread switching without having to save or reload a context of a thread, thereby saving processing time. A person of ordinary skill in the art would have recognized that multithreading and thread switching are beneficial tools for a system because when one thread stalls, another thread may be switched in and executed, thereby preventing the processor from going idle. With thread switching comes the switching of register files as well. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagami to toggle between active and inactive register file states because switching of register sets supports thread switching, which as described above, prevents a processor from staying idle during a stall, thereby increasing throughput and efficiency.
- d) at least one functional unit that executes a first program, said functional unit coupled to said second port of said register files, said functional unit responsive to commands exclusively involving architectural register operands that map onto to the registers within a register file that is in the active state. See Fig.1 and note that the first set of functional units would be component

5, which executes manipulation instructions like multiplication, addition, etc. In order to do this, it must be coupled to registers, as shown in Fig.1, for retrieving an operating upon operands.

e) a data assembly responsive to an instruction set comprising at least:

(i) a command that causes data to be moved between the DRAM array and a register file that is in the inactive state. See Parady, and note the thread switch command that from the abstract and from Fig.3. More specifically, when a thread switch occurs, the register file associated with the “switched-in” thread will become active and consequently used by the system during execution of that thread. At the same time, the register file associated with the “switched-out” thread will become inactive.

(ii) Inagami has taught a command to transfer data between the DRAM array and a data register file. See Fig.4. Neither Inagami nor Parady have taught such a transfer involves an inactive register file. However, Bissett has taught performing data prefetches (loads) in the background. And, this is beneficial because background operations do not influence software execution, thereby allowing the current program to run normally while also accomplishing a task in the background. See column 4, lines 3-8. And, as is known in the art, prefetching is beneficial because data is brought in from memory before an instruction needs it. Then, when the instruction does actually execute, the data is already fetched, allowing the instruction to execute more quickly. Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagami in view of Parady in view of Bissett such that Inagami in view of Parady performs background prefetching to a register file is not currently in use (one that corresponds to an inactive thread).

(iii) ***NOTE*** As an alternate interpretation, an inactive register file could be nothing more than a file which is not currently being accessed by the functional units. This could be the case in Fig.1, where there are many register files. Some may not be needed by functional units 5. These would be inactive and a load instruction may be the instructions used to transfer data to the inactive file. In addition, when the functional units do request operands from the inactive file (say in response to an add instruction), then the inactive file becomes active and accessible to the functional unit. Applicant should amend accordingly to avoid both interpretations of Inagami, etc.

54. Referring to claim 49, Inagami has taught a digital processor comprising:
- a) an embedded array having a plurality of random access memory cells arranged in rows and columns. See Fig.1 and note that main memory (component 1) inherently comprises rows and columns of memory cells. Also, note that the memory is embedded in the overall system (Inagami states that Fig.1 shows a vector processor and the memory is part of the processor). Inagami has not explicitly taught that the memory array is a DRAM array. However, Official Notice is taken that DRAM and its advantages are well known and expected in the art. More specifically, DRAM is a very popular memory technology because of its high density and low price (in comparison to other memory such as SRAM). Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagami's main storage to be a DRAM array.
 - b) first and second dual-port register files, each of said register files capable of parallel transferring data between a row of said embedded DRAM array. See Fig.1, components VMR0-VMR7 and VR0-VR7. Note that these registers may be loaded or stored in response to a single

load or store signal shown in Fig.4, Fig.5, and column 1, lines 52-64. Also, it should be noted that these files are at least dual-port in that they have ports for reading and writing (see Fig.1).

c) Inagami has not taught that each of said register files is also capable of being placed into an active state and an inactive state. However, Parady has taught a thread switch command which performs such state toggling. More specifically, in Parady, every time a thread switch occurs, a new register file becomes active and the previously active register file becomes inactive. This allows for thread switching without having to save or reload a context of a thread, thereby saving processing time. A person of ordinary skill in the art would have recognized that multithreading and thread switching are beneficial tools for a system because when one thread stalls, another thread may be switched in and executed, thereby preventing the processor from going idle. With thread switching comes the switching of register files as well. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagami to toggle between active and inactive register file states because switching of register sets supports thread switching, which as described above, prevents a processor from staying idle during a stall, thereby increasing throughput and efficiency.

d) first and second functional units. See Fig.1, components 5 and 2, respectively.

e) a method for processing data comprising:

- (i) manipulating data in a data register with a register file using said first functional unit that is in an active state. See column 4, lines 35-37.
- (ii) using said second functional unit. See column 4, lines 32-35.
- (iii) placing an inactive register file into said active state, whereby when the register set is activated, it becomes an architectural register set of said first functional unit. See Parady,

and note the thread switch command that from the abstract and from Fig.3. More specifically, when a thread switch occurs, the register file associated with the “switched-in” thread will become active and consequently used by the system during execution of that thread.

(ii) Inagami has taught unidirectionally transferring data between a row of said embedded DRAM array and a selected data register file. See Fig.4. Neither Inagami nor Parady have taught such a transfer involves an inactive register file. However, Bissett has taught performing data prefetches (loads) in the background. And, this is beneficial because background operations do not influence software execution, thereby allowing the current program to run normally while also accomplishing a task in the background. See column 4, lines 3-8. And, as is known in the art, prefetching is beneficial because data is brought in from memory before an instruction needs it. Then, when the instruction does actually execute, the data is already fetched, allowing the instruction to execute more quickly. Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagami in view of Parady in view of Bissett such that Inagami in view of Parady performs background prefetching to a register file is not currently in use (one that corresponds to an inactive thread).

(iii) ***NOTE*** As an alternate interpretation, an inactive register file could be nothing more than a file which is not currently being accessed by the functional units. This could be the case in Fig.1, where there are many register files. Some may not be needed by functional units 5. These would be inactive and a load instruction may be the instructions used to transfer data to the inactive file. In addition, when the functional units do request

operands from the inactive file (say in response to an add instruction), then the inactive file becomes active and accessible to the functional unit. Applicant should amend accordingly to avoid both interpretations of Inagami, etc.

55. Referring to claim 50, Inagami has taught a processor comprising:
- a) an embedded array comprising a plurality of random access memory cells arranged in rows and columns. See Fig.1 and note that main memory (component 1) inherently comprises rows and columns of memory cells. Also, note that the memory is embedded in the overall system (Inagami states that Fig.1 shows a vector processor and the memory is part of the processor). Inagami has not explicitly taught that the memory array is a DRAM array. However, Official Notice is taken that DRAM and its advantages are well known and expected in the art. More specifically, DRAM is a very popular memory technology because of its high density and low price (in comparison to other memory such as SRAM). Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagami's main storage to be a DRAM array.
 - b) an embedded row address register that holds a pointer that points to a row of the DRAM array. See Fig.1, component 60, and column 5, lines 13-17.
 - c) embedded first and second register files, each of said register files capable of being loaded or stored in response to a single latch signal. See Fig.1, components VMR0-VMR7 and VR0-VR7. Note that these registers may be loaded or stored in response to a single load or store signal shown in Fig.4, Fig.5, and column 1, lines 52-64.
 - d) Inagami has not taught that each of said register files is capable of being placed into an active state and an inactive state. However, Parady has taught a thread switch command which

performs such state toggling. More specifically, in Parady, every time a thread switch occurs, a new register file becomes active and the previously active register file becomes inactive. This allows for thread switching without having to save or reload a context of a thread, thereby saving processing time. A person of ordinary skill in the art would have recognized that multithreading and thread switching are beneficial tools for a system because when one thread stalls, another thread may be switched in and executed, thereby preventing the processor from going idle. With thread switching comes the switching of register files as well. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagami to toggle between active and inactive register file states because switching of register sets supports thread switching, which as described above, prevents a processor from staying idle during a stall, thereby increasing throughput and efficiency.

d) embedded first and second functional units, said first and second functional units having respective first and second instruction sets and capable of accessing said first and second register files while in said active state. See Fig.1 and note that the first set of functional units could be component 5, which executes manipulation instructions like multiplication, addition, etc. See column 4, lines 35-37. The second set of functional units could be component 2 of Fig.1, which executes data transfer instructions such as loads and stores. See column 4, lines 32-35.

e) wherein said first and second register files comprise a parallel access port (see connections between register files and switches 120-123) operative to parallelly transfer contents of one of said register files between a DRAM row as selected by said row-address register (again see Fig.1 and note that these connections allow register files to be loaded with DRAM contents), each said register file further comprising at least a second access port operative to transfer data between a

selected register file and said second functional unit (see the connections in Fig.1 between the switches 120-123 and the load/store pipes (second functional unit...these allow data to be read from the register files by the pipes and ultimately stored in DRAM).

f) wherein said first instruction set comprises at least:

(i) a command to manipulate data in a data register with a register file. See column 4, lines 35-37.

g) wherein said second instruction set comprises at least:

(i) a command to perform arithmetic on said row address register. See Fig.6, column 7, line 66, to column 8, line 9, and note that arithmetic is performed on the row address registers by the start address calculation unit 310.

(ii) a command to load the entire row pointed to by said row address register into a selected set of registers of said register files. See Fig.4, and column 1, lines 52-59. Note that each row of the memory shown in Fig.4 which corresponds to a mask bit equal to 1 is transferred to a register file. For instance, a0, which is clearly the first row in the main storage, is transferred to a register file. The data a0 is the entire row data. Inagami in view of Parady has not taught that the selected set of said register files are to be loaded in the inactive state. However, Bissett has taught performing data prefetches (loads) in the background. And, this is beneficial because background operations do not influence software execution, thereby allowing the current program to run normally while also accomplishing a task in the background. See column 4, lines 3-8. And, as is known in the art, prefetching is beneficial because data is brought in from memory before an instruction needs it. Then, when the instruction does actually execute, the data is already

fetched, allowing the instruction to execute more quickly. Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagami in view of Parady in view of Bissett such that Inagami in view of Parady performs background prefetching to a register file is not currently in use (one that corresponds to an inactive thread).

Response to Arguments

56. On page 24 of the remarks, applicant argues that “one cannot substitute DRAM for the SRAM or other memory types shown in the prior art cited by the examiner as the primary bases for his rejections.” The examiner asserts that Inagami does not disclose the type of memory used in the system. Therefore, the examiner is not substituting DRAM for SRAM or any other memory type. The examiner has therefore stated that it would be obvious to have Inagami’s memory be a DRAM because of its low cost and high density.

57. On page 26 of the remarks, applicant argues that “the examiner provide explicit references that teach an embedded DRAM processor with the functionality of the claimed inventions.” For example, applicant “requests that the examiner provide explicit references that teach register “move” functionality in the context of an embedded DRAM processor.” The examiner asserts that the reference provided is already a satisfactory teaching for two reasons:
a) Applicant is reading “embedded DRAM processor” too narrowly. Dictionary.com defines “embedded” in the following way: “To cause to be an integral part of a surrounding whole.” Clearly, in Fig.1 of Inagami, the memory is an integral part of the overall processing system (processor). Hence, Inagami teaches an embedded DRAM processor.

b) As stated by the examiner, a move instruction is a well known processor instruction. It is not limited to a certain context, such as in an embedded DRAM processor. In fact, a move instruction has very little to do with the DRAM itself since a move is associated with registers, which appear in the majority of processors nowadays. A move allows for the moving of data from one register to another (i.e., copying) or the moving of an immediate value into a register. Move instructions are essential instructions in a processor and therefore, it would have been obvious to implement such an instruction in Inagami, regardless of the type of processor Inagami has taught.

58. On page 27, regarding the number of times Official Notice was taken by the examiner, applicant argues that “Each claim is a different invention and hence the use of Official Notice in each instance is unique. The examiner focuses on numerology, but instead should focus on the fact that every independent claim presented by applicant is rejected based on Official Notice. Applicant’s issue with the examiner’s application of Official Notice is not the number of times it is used per se, but rather the number of times it is used as a basis of rejecting highly varying and different inventions...With all due respect, everything appears obvious to the examiner, since he can always cite Official Notice to fill in any such holes left by the prior art.” In response, while the examiner agrees that each claim is a different invention, it is not clear how this precludes the use Official Notice. The examiner is not clear why repeating Official Notice for different claims is unacceptable. If the examiner could only apply Official Notice for concept A in a first claim but not for concept A in a second claim, then the purpose of Official Notice is defeated. That is, if the examiner had to go out and find prior art for the teaching of concept A for the second claim, then why use Official Notice at all? Why not just use the prior art for the first claim as

well? In addition, the examiner is filling holes with Official Notice on some occasions. However, the idea of a 103 rejection is to fill holes left by the prior art. Official Notice has only been taken on concepts (the holes) which the examiner deems to be well known. When previously challenged by applicant to show that these concepts were well known, prior art was cited to provide such a showing.

59. All of the remaining arguments (directed towards the 103 rejections) appear to be copies of arguments from the previous amendment filed on December 27, 2005. Consequently, the examiner responds to those arguments in the same manner as in the Office Action mailed on

60. Regarding applicant's arguments for claims 1, 13, 46, and 48:

a) Applicant argues on page 28, lines 11-23, that the prior art's row address register is different from applicant's for a number of reasons, including that the row address register may be used in precharge-row and deactivate-row instructions. The examiner asserts that a row address register in the prior art is used to hold addresses of memory which is to be accessed. Therefore, the register holds an address of memory from which data is to be loaded (read). In order to read/load data from memory, the memory must be precharged. Consequently, if address X is to be read from, then memory location X in memory must be precharged. As a result, the row address register would be used to precharge and deactivate rows (because the row which the row address register points to will need to be precharged and deactivated).

b) On page 29, lines 27-30, applicant argues that the SDRAM commands in Wright are hardware layer memory commands sent directly to the DRAM device, and hence Wright does not teach or suggest the use of the use of a row address register nor a precharge command nor a deactivate

command as taught by Applicant. However, the examiner asserts that Inagami has already taught the row address register and Wright was added to Inagami to show that precharging/deactivating must be done. More specifically, precharging is only done when a row is to be read. Therefore, the row address must be known in order to precharge it. In Inagami, the row address register holds the address to be read/precharged. Without the row address register, how would the system know which memory row to access? In addition, actions occur in response to commands. It is not clear to the examiner how there is no precharge command in Wright when the cited passages explicitly refer to commands.

c) Regarding claims 46 and 48, applicant argues on page 29, lines 20-24, that the amendments now specify that loading is performed in response to an instruction issued after the precharging. However, applicant's attention is drawn to Wright, column 1, lines 43-45, and note that separate commands are required for precharging and accessing rows of memory. Therefore, a precharge would occur and then the actual load would follow.

d) Regarding claim 13, applicant's arguments on page 29, lines 6-12, are similar to those for claim 48. Consequently, the response to the arguments for claim 48 (part (c) above) also applies to the arguments for claim 13.

61. Regarding applicant's arguments for claims 2-5, 11, and 26-27:

a) Applicant argues on page 30, lines 12-18, that "Parady teaches the use of threads which are assigned sets of registers in which to perform the associated processing tasks. However, the threads do not correspond to any particular function element such as component 38 or 40 or Fig 1. The threads are simply different sets of software executed in round robin fashion. Thus any

component in Fig.1 could by used by any thread at a given time.” While it is true that multiple threads assigned to multiple register files take turns executing in Parady, it is not clear why each thread must correspond to a function unit as argued by applicant. When a thread executes, it does correspond to at least one set of the functional units, since those functional units will be executing that thread’s instruction.

b) On page 31, lines 1-6, applicant argues that “Any attempt to combine Parady with Inagami would be inoperable as the use of completely independent threads of Parady would make the vector processing of Inagami impossible. The threads lack the organization and structure necessary for any sort of vector process.” This argument, while fully considered, is not persuasive because the examiner does not believe that Parady’s system precludes vector processing. More specifically, vector processing operations, as taught by Inagami, may be broken up into threads, where each thread would operate according to the rules of Inagami. As a result, the advantages of both systems are achieved.

c) Applicant argues on page 31, lines 16-24, that “there would be no motivation to combine Parady with Inagami because the vector processor does not need to do thread switching and also the thread switching has nothing to do with the claimed invention.” While the claimed invention may have nothing to do with thread switching, it contains limitations which are found in thread switching systems. And, as shown in Parady, thread switching has advantages. Consequently, by modifying Inagami to include thread-switching, the advantages of Parady are achieved.

62. Regarding applicant’s arguments for claim 16:

Art Unit: 2183

a) Applicant argues on page 33, lines 8-12, that “These limitations of Claim 16 set forth far more than just the general concept of performing a data prefetch as disclosed in Bissett. They include (i) a command to load a set of selected elements of a row pointed to by a row address, and (ii) that the selected set of data registers is inactive, and determined by a bit mask. This is far more than just a general prefetching operation.” The examiner does note that Bissett does just teach general prefetching, but Inagami already teaches the details of the claim 16 limitations in question. Specifically, Inagami has already taught that when data is fetched from memory, the row register is used. Consequently, if prefetching were added, the row register would still be used.

Conclusion

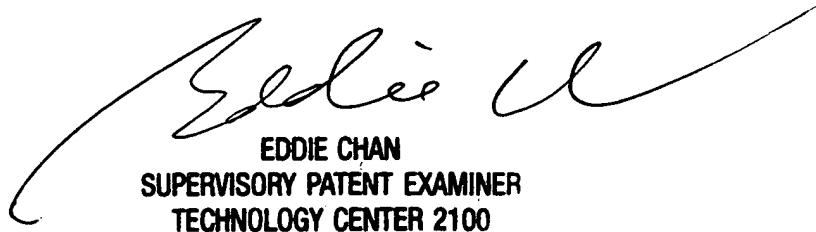
Any inquiry concerning this communication or earlier communications from the examiner should be directed to David J. Huisman whose telephone number is (571) 272-4168. The examiner can normally be reached on Monday-Friday (8:00-4:30).

If attempts to reach the examiner by telephone are unsuccessful, the examiner’s supervisor, Eddie Chan can be reached on (571) 272-4162. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Art Unit: 2183

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

DJH
David J. Huisman
March 27, 2006



EDDIE CHAN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100